Modellgetriebene ad-hoc Integration von Logistikdienstleistern – Integrationsansatz und Prototyp

Robert Kunkel
André Ludwig
Bogdan Franczyk

Veröffentlicht in:
Multikonferenz Wirtschaftsinformatik 2012
Tagungsband der MKWI 2012
Hrsg.: Dirk Christian Mattfeld; Susanne Robra-Bissantz

Braunschweig: Institut für Wirtschaftsinformatik, 2012

Modellgetriebene ad-hoc Integration von Logistikdienstleistern – Integrationsansatz und Prototyp

Robert Kunkel, André Ludwig, Bogdan Franczyk

Universität Leipzig, Institut für Wirtschaftsinformatik, Grimmaische Str. 12, 04109 Leipzig,
E-Mail: {kunkel, ludwig, franczyk}@wifa.uni-leipzig.de

Abstract

Der Logistikdienstleistungssektor ist durch arbeitsteilige sowie kurz-, mittel- und langfristige Zusammenarbeit gekennzeichnet. Insbesondere Fourth Party Logistics (4PL) stehen permanent vor der Aufgabe unterschiedliche Logistikdienstleister und damit auch deren Informationssysteme ad-hoc und medienbruchfrei in unternehmensübergreifende Informationsflüsse zu integrieren. Dieser Beitrag stellt verschiedene logistikspezifische Integrationsvarianten, einen modellgetriebenen Integrationsansatz sowie ein Lösungskonzept auf Basis der Logistik Service Engineering & Management (LSEM)-Plattform vor. Die Umsetzung eines hierfür entwickelten Prototyps veranschaulicht das Lösungskonzept.

1 Einleitung

Das Managementkonzept und Unternehmensmodell Fourth-Party Logistics (4PL) wurde 1996 vom Beratungsunternehmen Accenture erstmals benannt und definiert. Ein Logistikdienstleister vom Typ 4PL fasst Dienstleistungsangebote verschiedener anderer Logistikdienstleister (LD) zu einer komplexen Gesamtdienstleistung zusammen und koordiniert die Arbeit in den Lieferketten, ohne eigene logistische Ressourcen wie Fuhrpark oder Lagerhaus zu besitzen (siehe [2]). Ein 4PL stellt seine angebotenen Dienstleistungen somit ausschließlich aus den Angeboten von sogenannten Second und Third Party Logistics (ausführende und spezialisierte Logistikunternehmen) zusammen. Als Integrator und Koordinator von Lieferketten stehen dem 4PL ausschließlich Informationssysteme und logistisches Domänenwissen zur Verfügung.

Informationen und der mit ihnen verbundene Informationsfluss sind ein wesentlicher Teil der Logistik. Durch die effiziente Bereitstellung von Informationen zur richtigen Zeit kann der Güterfluss in einer Lieferkette deutlich beschleunigt werden (siehe [14]). In sämtlichen Phasen der Zusammenarbeit, ob Verhandlung, Erstellung und Erbringung von Dienstleistungen oder bei der Abrechnung, müssen Informationen zwischen den LD und dem 4PL ausgetauscht werden. Alle anfallenden Informationen müssen dabei zur Planung, Steuerung und Kontrolle der Lieferkette vom 4PL verarbeitet werden. Um diese Informationsflut beherrschen zu können, wird ein Softwaresystem benötigt, welches die anfallenden

Informationen sammelt, aggregiert und verarbeitet. Eine mögliche Lösung dieses Problems wird durch die Logistik Service Engineering and Management (LSEM)- Plattform bereitgestellt, welche sich an den Anforderungen des Geschäfts- und des Softwarelevels orientiert (siehe [11]). Integrationsplattformen, wie die LSEM-Plattform, bieten dem 4PL die Möglichkeit eigene Anwendungssysteme und die seiner Partner einzubinden, um die anfallenden Informationen effizient, performant und über Regeln automatisierbar zu verarbeiten. Im Gegensatz zur innerbetrieblichen Integration oder der Integration zwischen langfristig kooperierenden Unternehmen, erfordern die Spezifika der Logistikbranche verschiedene Arten der Integration. Neben der Teilnahme an mittel- bis langfristigen Kontrakten, muss es den LD ermöglicht werden, sich auf Basis einzelner Aufträge an den Lieferketten des 4PL zu beteiligen. Zusätzlich verbessert dies einerseits die Akzeptanz der Plattform bei LD und andererseits vergrößert sich die Gruppe der zur Verfügung stehenden LD für den 4PL. Zur Integration von Anwendungssystemen der LD in die Plattform des 4PL müssen neben der Vollintegration (Vollintegration bezeichnet in diesem Beitrag die komplette Integration sämtlicher benötigter Funktionalitäten eines Anwendungssystems) weitere Integrationsarten, welche sich einerseits im Umfang und andererseits im zeitlichen Aufwand zur Einrichtung unterscheiden, zur Verfügung stehen (siehe [13]).

In diesem Beitrag werden die Spezifika der Integration von LD und deren Anwendungssysteme in eine logistische Plattform betrachtet. Kapitel 2 adressiert den Integrationsansatz, die zugrunde liegende Architektur der LSEM-Plattform, logistikspezifische Integrationsvarianten und die Umsetzung der Varianten mit Hilfe modellgetriebener Verfahren. Die prototypische Umsetzung des Integrationsansatzes wird in Kapitel 3 beschrieben. Anschließend werden grundlegende und verwandte Arbeiten betrachtet, während in Kapitel 5 ein kurzer Ausblick gegeben wird.

2 Integrationsansatz

Dem 4PL als Integrator von Lieferketten stehen ausschließlich Informationssysteme und logistisches Domänenwissen zur Verfügung, um komplexe Gesamtdienstleistungen von verschiedenen LD zusammenzustellen und die Arbeit in den Lieferketten zu koordinieren. Im nachfolgenden Abschnitt wird die grundlegende Architektur des Integrationssystems vorgestellt; Der Abschnitt 2.1 beschreibt den Aufbau der Integrationsplattform im Detail. Abschnitt 2.2 stellt logistikdomänenspezifische Integrationsarten vor, während im Abschnitt 2.3 auf die Umsetzung dieser durch modellgetriebene Verfahren eingegangen wird.

2.1 Architektur

Die LSEM-Plattform stellt eine Integrationsplattform für einen 4PL dar. Sie wurde innerhalb des Forschungsprojekts Logistik Service Bus (siehe [8]) entwickelt und baut auf dem Paradigma der Service-Orientierung auf, das sowohl auf geschäftlicher als auch auf technischer Ebene angewendet wird. Grundsätzlich gliedert sich die LSEM-Plattform in zwei Dimensionen. Zum einen umfasst die Laufzeitumgebung grundlegende Komponenten für die Bereitstellung und die Integration von Softwaresystemen und erlaubt auf dieser Basis die Komposition von Informationsflüssen und das Erstellen neuer Anwendungen. Zum anderen enthält die Werkzeugumgebung Komponenten, die das Konfigurieren und das Arbeiten mit der Laufzeitumgebung ermöglichen. Diese Systeme erlauben dem 4PL die Planung,

Steuerung und Kontrolle der Lieferketten. Die in Bild 1 dargestellte Architektur der LSEM-Plattform baut auf der S3-Referenzarchitektur (siehe [1]) auf.

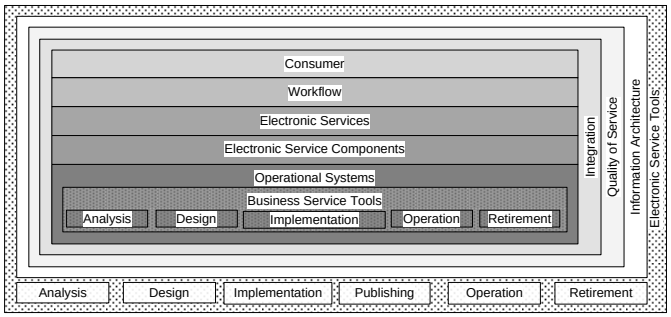

Bild 1: Die Architektur der LSEM-Plattform (aufbauend auf [1])

Die vorliegende Arbeit bezieht sich auf die Schichten der zweiten Dimension, welche grundlegende Funktionalitäten für die Erstellung von Integrationskomponenten bereitstellen. Hierbei werden die Integration Schicht, welche typische Enterprise Service Bus Eigenschaften umfasst und somit zentrale Komponenten für das Bereitstellen und Integrieren von Anwendungen bereitstellt, und die Information Architecture Schicht, welche ein kanonisches Datenformat für den Datenaustausch auf der Plattform definiert, benötigt. Eine ausführlichere Beschreibung der LSEM-Plattform und deren Schichten befindet sich in [11].

2.2 Integrationsvarianten

Die Zusammenarbeit des 4PL mit verschiedenen LD bedingt, wie in den vorherigen Kapiteln dieses Beitrages dargestellt, die Integration von Anwendungssystemen der LD in die zentrale LSEM-Plattform. Im Gegensatz zur innerbetrieblichen Integration oder der Integration zwischen langfristig kooperierenden Unternehmen, erfordern die Spezifika der Logistikbranche verschiedene Arten der Integration. Neben der Teilnahme an mittel- bis langfristigen Kontrakten, muss es einem LD ermöglicht werden, sich auf Basis einzelner Aufträge an den Lieferketten des 4PL zu beteiligen. Weiterhin wird zur Integration der Anwendungssysteme der LD eine Integrationsumgebung benötigt, welche die Integration nach adaptiven Verfahren (siehe [18] und [7]) ermöglicht. Die Adaption ermöglicht dabei eine nicht-invasive Integration, welche die innere Struktur des Systems und die vorhandenen Schnittstellen nicht verändert. Hierdurch müssen die zu integrierenden Anwendungssysteme nicht angepasst werden.

Zur schrittweisen Integration von Anwendungssystemen der LD wird somit eine Integrationsumgebung benötigt, welche sowohl die unterschiedlichen Integrationsvarianten unterstützt als auch die Anwendungssysteme durch ein adaptives Integrationsverfahren mit Schnittstellen ausstattet. Diese Integrationsumgebung muss hierbei individuell für jeden LD erstellt, konfiguriert und bereitgestellt werden und übernimmt anschließend sämtliche Kommunikationsaufgaben zwischen dem LD und dem 4PL. Die hierzu benötigten Komponenten werden entweder durch die LSEM-Plattform bereitgestellt oder durch diese entwickelt. Durch die zweite Dimension der LSEM-Plattform, welche für die Bereitstellung und die Integration von Softwaresystemen zuständig ist, werden Integrationsumgebungen entwickelt, welche sich in die erste Dimension der LSEM-Plattform einordnen und für die zu integrierenden Anwendungssysteme Integrationslogik und Services erzeugen. Nähere

Erläuterungen zur Umsetzung und zur theoretischen Konzeption der Komponenten sind [11], [10] und [7] zu entnehmen. Daten und Informationen können in jedem Anwendungssystem in unterschiedlichen, oftmals proprietären Formaten und Modellen vorliegen. Zur Lösung dieses Problems werden in der Integrationsumgebung *Message Translator* eingesetzt, welche das vorliegende Format der LD in das kanonische Format der LSEM-Plattform und zurück übersetzen. Für Anwendungssysteme sollte transparent bleiben, welches Ziel eine gesendete Nachricht hat und über welche Stationen, z.B. Message Translator, sie gesendet wird. Diese Entkopplung von Sender und Empfänger wird über *Message Channel* erreicht. Zur Weiterleitung von Nachrichten aufgrund des Inhaltes werden *Content-based Router* eingesetzt, welche über definierte Regeln den Empfänger bestimmen.

Ziel dieses Integrationsvorgehens ist die Integration von verschiedenen LD an die Logistikplattform des 4PL mit Hilfe einer Integrationsplattform. Der Einsatz einer solchen Integrationsplattform ermöglicht es dem 4PL über einheitliche Services mit den LD zu kommunizieren. Hierzu werden auf der LSEM-Plattform für verschiedene Rollen (z.B. Lager, Transport und Umschlag) abstrakte Services definiert, welche durch die LD implementiert und bereitgestellt werden müssen. Weiterhin können für die Services Übertragungstypen und Kommunikationseinstellungen festgelegt werden. Die implementierten Services der Integrationsumgebung können nun von der LSEM-Plattform aufgerufen werden und die Services der LSEM-Plattform können durch die Integrationsplattform genutzt werden. Dies ermöglicht das Umsetzen von One-Way - (z.B. dem Senden von Avis in beide Richtungen) und Request-response - Operations (z.B. Anfrage Transportauftrag). Die ausführliche Beschreibung von Rollen, abstrakten Services, Typen und Kommunikationseinstellungen erfolgt in Abschnitt 2.3. In Bild 2 sind drei Integrationsvarianten dargestellt, bei welchen sich von links nach rechts jeweils der Integrations- und Automatisierungsgrad, jedoch auch der Integrationsaufwand erhöhen. Die Varianten basieren auf der Arbeit von [13] und erweitern diese um für diese Arbeit benötigte Integrationskonzepte. Bei allen drei Varianten übernimmt die Integrationsumgebung die Implementierung und Bereitstellung der abstrakten Services sowie die Kommunikation mit der LSEM-Plattform über Message Channel. Variante I stellt den ersten Schritt im Integrationsprozess dar. Die Kommunikation und der Informationsaustausch werden vollständig von der Integrationsumgebung übernommen. Die Integrationsvariante I soll dem LD ad-hoc und ohne weitere Integrationsaufwände und Integrationskosten bereitgestellt werden. Diese Variante ermöglicht es den LD sich kurzfristig an Kontrakten des 4PL zu beteiligen ohne Kosten für die adaptive Integration seiner Anwendungssysteme aufwenden zu müssen. Zur Interaktion zwischen LD und der Integrationsumgebung steht ein Rich Client zur Verfügung, welcher die Kommunikation von Menschen mit den Services der Integrationsumgebung durch Human-Service Interaction (HSI) ermöglicht (siehe[16] und [6]). Sämtliche Serviceaufrufe werden dem LD als Aufgaben in menschenlesbarer Form bereitgestellt.

Die in Variante I bereitgestellte Integrationsinfrastruktur ermöglicht es nun in den weiteren Schritten die Anwendungssysteme des LD schrittweise zu integrieren. Neben der kurzfristigen Zusammenarbeit können die LD bei mittel- bis langfristigen Kooperationen den Integrationsgrad erhöhen und somit den Informationsaustausch schrittweise automatisieren. Hierzu können, wie in Variante II dargestellt, einzelne Komponenten des Anwendungssystems adaptiert und die so gewonnenen Schnittstellen mit der Integrationsumgebung integriert werden. Beispielsweise könnte in einem ersten Schritt die Übermittlung von

Auftragsanfragen automatisiert werden. Durch das Bereitstellen entsprechender Message Translator und Router werden die Informationen direkt in das Anwendungssystem übertragen. Der Rich Client übernimmt für die integrierten Services anschließend Monitoringfunktionalitäten, für nicht integrierte Services wird weiterhin die Kommunikation über HSI zur Verfügung gestellt.

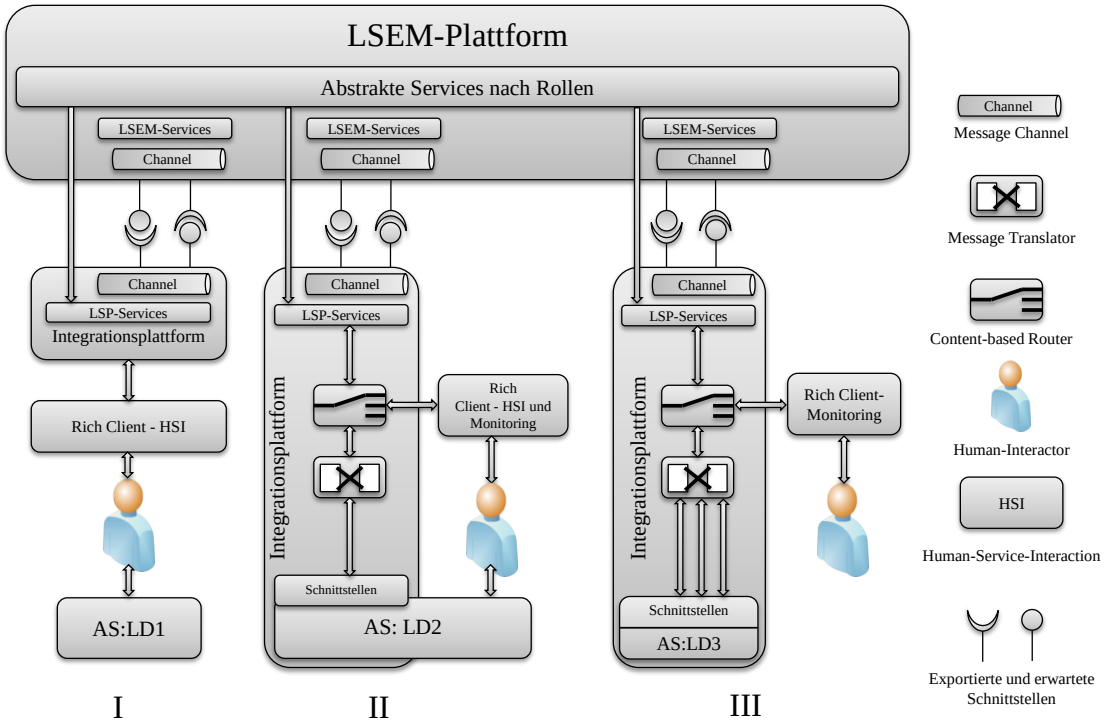

Bild 2: Integrationsvarianten (in Anlehnung an [13])

Die Variante III stellt die Vollintegration des Anwendungssystems dar. Sämtliche Services sind hierbei über entsprechende Message Channel, Translator und Router mit den für das Anwendungssystem bereitgestellten Schnittstellen verbunden. Der Rich Client übernimmt hierbei ausschließlich das Monitoring der Integrationsumgebung. LD können mit Hilfe der dargestellten Varianten je nach Art der Zusammenarbeit über den Integrationsgrad entscheiden, während der Integrationsfortschritt für den 4PL transparent ist, da bei jeder Variante die gleichen Schnittstellen der Plattform unterstützt werden.

2.3 Modellgetriebene ad-hoc Integration

Die im vorhergehenden Kapitel dargestellten Integrationsarten ermöglichen einem 4PL und den LD, neben mittel- und langfristigen Kontrakten, die Zusammenarbeit in kurzfristigen Kontrakten. Die ad-hoc Zusammenarbeit wird durch die effiziente Bereitstellung der Integrationsumgebung und die Serviceimplementierung durch HSI in Integrationsvariante I ermöglicht, ohne dass für den LD weitere Integrationsaufwände und -kosten entstehen. Für diese effiziente ad-hoc Bereitstellung einer Integrationsumgebung eignen sich vor allem modellgetriebene Verfahren, da diese besondere Stärken aufweisen, wenn mehrere verwandte Produkte entwickelt werden und Teile sich wiederverwenden lassen (siehe [17] und [18]). Mit Hilfe von modellgetriebenen Ansätzen lässt sich die Gesamt-komplexität dieses Integrationsansatzes beherrschen, da Modelle das System auf einer

höheren Abstraktionsstufe beschreiben und sich die Komplexität somit auf das Modell selbst (Modellierung durch den 4PL als Logistikdomänenexperte) und auf die Abbildung des Modells auf die Zielsprache, also die Generierung und Interpretation (Modelltransformationen innerhalb der LSEM-Plattform), aufteilen lässt. Beide Teile können getrennt voneinander bearbeitet werden, wodurch beim Modellieren keine technischen Details der Implementierung bekannt sein müssen [17].

Die nächsten Abschnitte beschreiben das modellgetriebene Vorgehen und gliedern sich in einen Abschnitt zur Modellierung durch Logistikdomänenexperten als Grundlage zur Integration und der Modelltransformation, im speziellen der Generierung der Integrationsumgebung durch modellgetriebene Verfahren.

2.3.1 Modellierung als Grundlage zur Integration

Als Ausgangspunkt zur Modellgetriebenen Integration werden vom Domänenexperten, in diesem Fall dem 4PL, die in Bild 3 dargestellten Modelle beschrieben. Diese gliedern sich in die vier Teilmodelle Rolle, abstrakter Service, Typen und Konfiguration. Die Definition dieser Modelle ermöglicht es innerhalb der LSEM-Plattform Prozesse zu erstellen und darin (abstrakte) Services der LD zu nutzen, ohne sich bereits auf einen ausführenden LD festgelegen zu müssen. Dies wird ermöglicht da jeder beteiligte LD mit Hilfe der Integrationsumgebung die gleichen für ihn definierten Services implementiert.

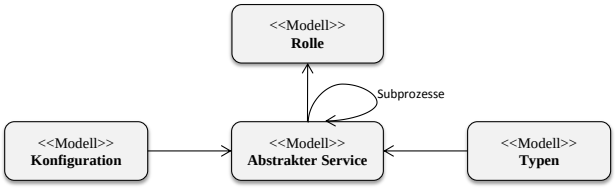

Bild 3: LSEM-Integrationsmodell

Im ersten Schritt legt der 4PL Rollen (z.B. Transport- oder Lageranbieter) fest. Diese dienen ausschließlich der Zuordnung und Zusammenfassung von Services, welche durch einen LD implementiert werden müssen. Die Definition eines abstrakten Services enthält die zu implementierenden Serviceoperationen. Für die einzelnen Operationen werden die zu übertragenden Datentypen ausgewählt oder neu modelliert. Die einzelnen Typen werden hierbei mit Hilfe der Extensible Markup Language (XML) beschrieben und können in verschiedenen abstrakten Services wiederverwendet werden. Im Konfigurationsmodell werden Kommunikationsparameter definiert. Beispielsweise werden hier synchrone und asynchrone Aufrufe und Webservice-Kommunikationsarten wie z.B. One-Way-, Request-Response-, Solicit-Response- und Notification unterschieden (siehe [19]).

2.3.2 Modelltransformation zur Generierung der Integrationsumgebung

Zur ad-hoc Bereitstellung der Integrationsumgebung werden Modelltransformationen auf Basis des in Bild 3 dargestellten LSEM-Integrationsmodells verwendet. Die auf der LSEM-Plattform bereitgestellten Modelle können nun durch modellgetriebene Verfahren in der Integrationsumgebung verwendet werden, um einerseits die Quellmodelle in konkretere Zielintegrationsmodelle (in Bild 4 im unteren Bereich mit <<Modell>> dargestellt) zu

transformieren und andererseits durch Interpretation und Generierung lauffähigen Code zu erzeugen (in Bild 4 im unteren Bereich mit <<generiert>> dargestellt).

Nach der Auswahl der zu konkretisierenden abstrakten Services über das Rollenkonzept werden in der Integrationsumgebung konkrete Services mit wohldefinierten Schnittstellen und Typen erzeugt. Die so generierten Service-Instanzen (konkreter Service) sind durch die LSEM-Plattform nach der Registrierung aufrufbar und können zur Laufzeit auf der LSEM-Plattform die abstrakten Services, welche in den Prozessen des 4PL verwendet wurden, ersetzen und die Prozesse somit in konkrete, lauffähige Prozesse umwandeln.

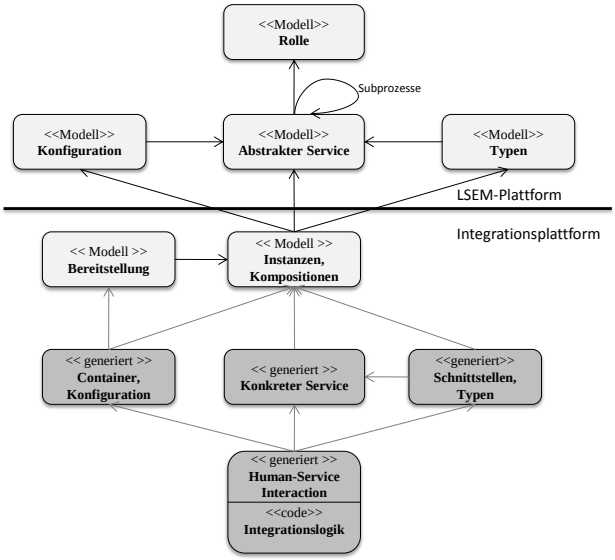

Bild 4: Quell- und Zielintegrationsmodelle

Als erster Schritt werden dabei in Abhängigkeit zur gewählten Rolle ein oder mehrere abstrakte Services mit dazugehörigem Typenmodell ausgewählt und in ein Instanzen- und Kompositionsmodell überführt. Für jeden abstrakten Service wird das definierte Konfigurationsmodell in ein Bereitstellungsmodell innerhalb der Integrationsumgebung überführt. Diese Modelle definieren die zu generierenden Services und deren Bereitstellung innerhalb der Integrationsumgebung. Somit können anschließend die konkreten Services, deren Schnittstellen und Typen generiert und mit Hilfe der erstellten Konfigurationen innerhalb von Service-Containern bereitgestellt werden. Das vorgestellte Integrationsmodell unterstützt alle Arten der in Bild 2 vorgestellten Integrationsvarianten. Zur Anbindung an die Plattform des 4PL steht somit eine Integrationsumgebung zur Verfügung, welche nach der Bereitstellung sämtliche Kommunikation über definierte Services zwischen dem 4PL und den LD übernimmt, wobei noch keinerlei Integration mit den Anwendungssystemen der LD stattfand. Um die ad-hoc zur Verfügung stehende Integrationsvariante I umsetzen zu können werden daher, wie in Bild 4 dargestellt, von den konkreten Services, deren Schnittstellen und Typen sowie der erzeugten Konfiguration eine Serviceimplementierung in Form einer Human-Service Interaction generiert. Die Integrationsumgebung mit der Integrationsvariante I kann durch die eingesetzten modellgetriebenen Verfahren unmittelbar zur Verfügung gestellt werden und erlaubt die Interaktion mit den erzeugten Services und die Darstellung der übertragenen Informationen. Um die Varianten II und III im weiteren Verlauf der Zusammenarbeit des 4PL mit dem LD zu unterstützen, werden einzelne Human-Service

Interactions durch Integrationslogik ersetzt. Diese Integrationslogik beinhaltet die manuelle Adaption des Anwendungssystems zur Erzeugung von Schnittstellen sowie die manuelle Implementierung der benötigten Message Translator und Router. Durch das schrittweise Ersetzen der Human-Service Interactions durch Integrationslogik kann das Anwendungssystem nach und nach integriert werden.

3 Prototyp

Das folgende Kapitel zeigt die prototypische Umsetzung des vorgestellten Integrationsansatzes. Hierbei soll die automatisierte Erstellung der Integrationsumgebung sowie die Umsetzung der Integrationsvariante I mit Hilfe von modellgetriebenen und generativen Ansätzen im Vordergrund stehen.

Zur Implementierung wurden die Werkzeuge Java Emitter Templates, Xtext und Xpand des Eclipse Modeling Projects (EMP), zur Erstellung und Transformation von Modellen, eingesetzt. Die Modellierungsumgebung für den 4PL wurde mit Hilfe von Eclipse RCP und Eclipse GMP bereitgestellt. Die Zielplattform bildet das jBPM-Projekt, welches das Grundgerüst zur Ausführung der generierten Human-Service Interactions (in jBPM Human-Task als Implementierung des OASIS-Standards WS-HumanTask) bereitstellt und durch das Hosting über den JBoss Application Server weitere Java Enterprise Edition - Services bietet. Die im Abschnitt 2.3.1 beschriebenen Modellierungswerkzeuge wurden als EMP-Projekt implementiert und sind als Eclipse-Plugin (RCP) lauffähig. Bild 5 zeigt den grafischen Modellierungseditor für das in Bild 3 dargestellte Typenmodell am Beispiel des Objektes *TransportOrderData*.

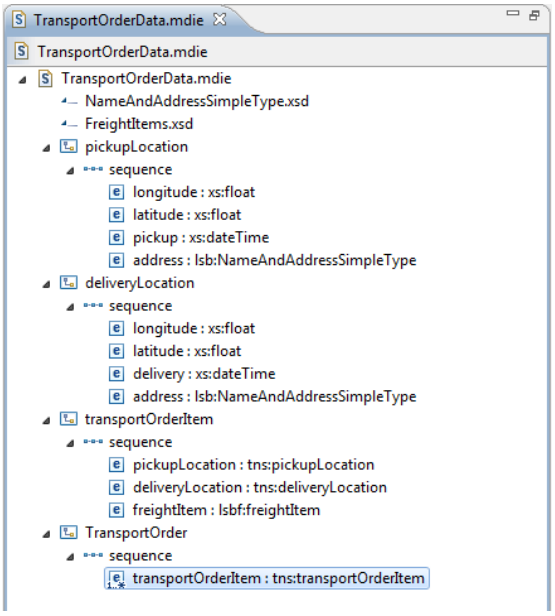

Bild 5: Modellierungstool für Datentypen am Beispiel von TransportOrderData

Wie in Bild 5 im oberen Bereich zu sehen, können einzelne Typenmodelle eine hierarchische Beziehung mit weiteren Typenmodellen aufbauen, indem diese als Untertypen verwendet werden. In den weiteren Editoren können jeweils Rollen, Konfigurationen und abstrakte Services angelegt, verwaltet und in Beziehung gesetzt werden. Auf Basis der erstellten

Quellmodelle können durch Modelltransformationen die Integrationsumgebung komplett generiert und die erforderlichen Services durch HSI automatisiert implementiert werden. Zur Modell zu Modell-Transformation (M2M) wurden das Xtend und Xtext-Framework eingesetzt. Hierbei werden die in Bild 4 gezeigten abstrakter Service- und Typenmodelle in ein plattformspezifisches Instanzen- und Kompositionsmodell und das Konfigurations- in ein plattformspezifisches Bereitstellungsmodell überführt.

Als Zielplattform für das Instanzen- und Kompositionsmodell dient das jBPM-Projekt, wohingegen das Bereitstellungsmodell als Zielplattform einen Jetty-Webserver nutzt. Aus diesen plattformspezifischen Zielmodellen kann nun durch eine Modell zu Text-Transformation (M2T), implementiert mit Hilfe des JET-Projektes, lauffähiger Quellcode generiert werden.

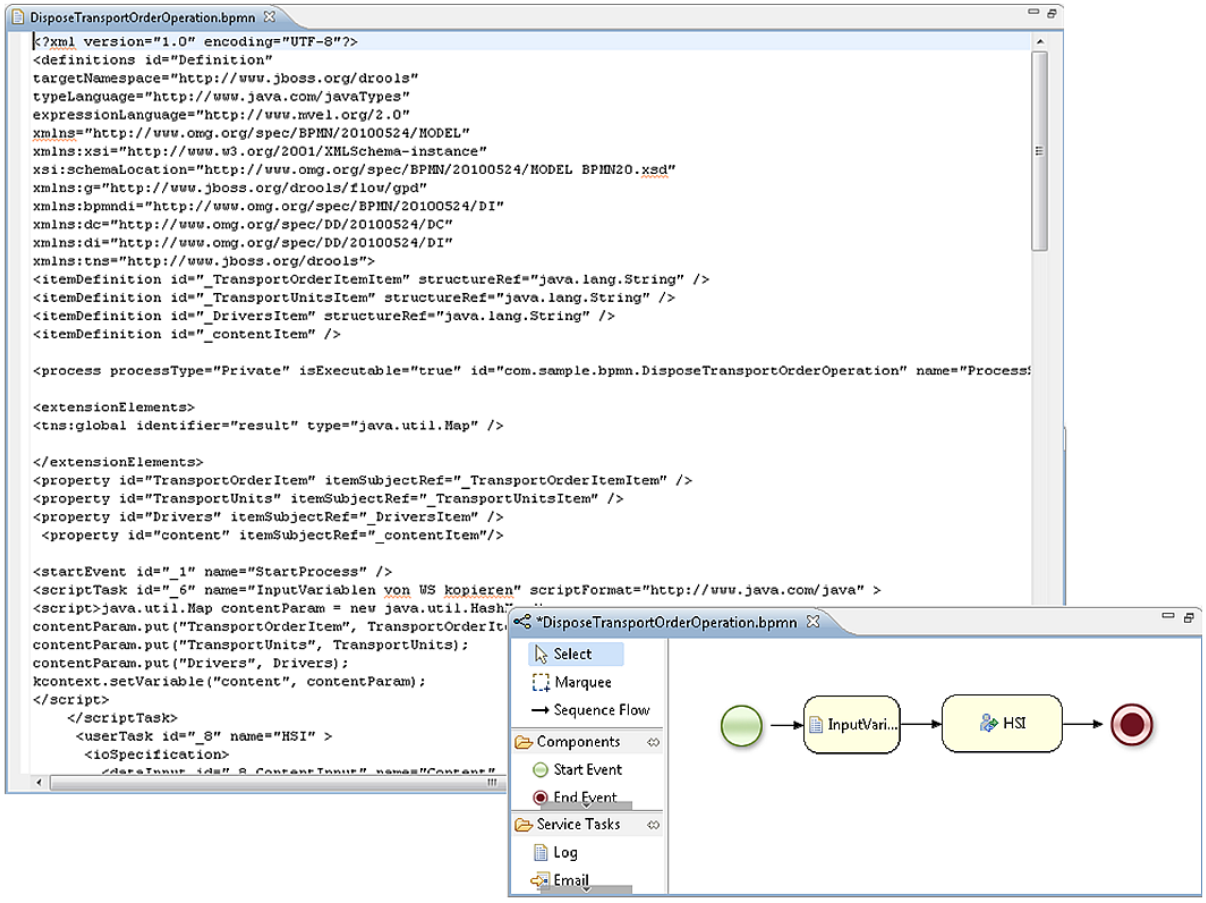

Bild 6: Generierter BPMN-Prozess in textueller und grafischer Repräsentation

Hierzu wird das Instanzen- und Kompositionsmodell in einen konkreten Service, welcher im Prototyp als WebService implementiert wurde, transformiert. Aus dem Bereitstellungsmodell werden Jetty-Service-Container inklusive Konfiguration generiert. Sämtliche generierten Services erhalten als Serviceimplementierung einen jBPM spezifischen Human Task. Jeder dieser Tasks besteht aus einem Business Process Model and Notation-Prozess (BPMN) und einer FTL-Datei (basierend auf der Hypertext Markup Language) zur grafischen Darstellung.

Die Bilder 6 und 7 zeigen einen generierten BPMN-Prozess in textueller und grafischer Repräsentation und die generierte Beschreibung eines jBPM-Human Task im FTL-Format.

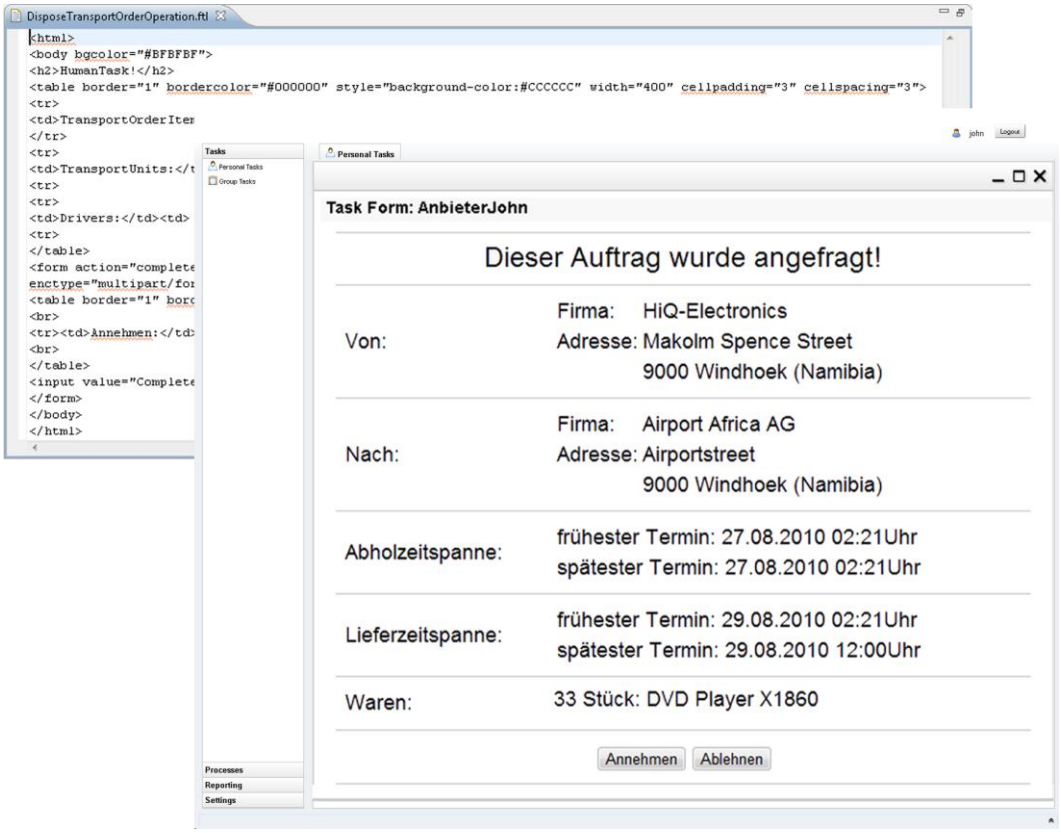

Bild 7: Beschreibung und Darstellung eines jBPM-Human Task im FTL-Format

4 Verwandte Arbeiten

Der beschriebene Ansatz baut auf den vier Themengebieten Integration Engineering, modelgetriebene Softwareentwicklung, Model-Driven Integration Engineering und Human-Service Interaction auf.

Bussler legt in seiner Arbeit [5] die Grundlagen der Geschäftsintegration. Aspekte wie Integrationskonzepte, elementare Integrationstypen und Standards werden diskutiert. Der Autor beschreibt die manuelle Integration, modellgetriebene Verfahren, zur Beschleunigung der Integration, werden nicht betrachtet. Benatallah et al. beschreibt in [3] Anforderungen der Adaption von Web Services und den Nutzen von Adaptern und Message Brokern zur Integration. Der Fokus wird von den Autoren auf Konversationsprotokolle gelegt, mit welchen beispielsweise die Aufrufreihenfolge von Serviceoperationen festgelegt werden kann. Neben dem Bilden von Schnittstellen in Altsystemen werden Aspekte der überbetrieblichen Integration und automatisierter Verfahren zum Erzeugen der Adapter nicht betrachtet. Gallas entwirft in [9] eine serviceorientierte IT-Architektur, welche durchgehend durch alle Phasen des Softwarezyklus unterstützt wird. Hierzu stellt er eine Integrationsschicht vor, welche den flexiblen Zugriff auf Softwarekomponenten ermöglichen soll. Der Schwerpunkt wird auf die innerbetriebliche Integration und eine Schicht zum Management von Services, bestehend aus einem Repository und einer Datendrehscheibe, gelegt. Der Autor fokussiert ausschließlich die Vollintegration, während der vorgestellte Ansatz eine schnell zur Verfügung stehende Integration über Human-Service Interactions betrachtet. Im Bereich der modellgetriebenen Softwareentwicklung ist vor allem die Arbeit von Stahl et al. [17] hervorzuheben, welche Techniken sowie Engineering- und Managementaspekte

modellgetriebener Ansätze adressiert. Diese Arbeit bildet eine Grundlage für den in diesem Beitrag vorgestellten Ansatz, indem die Aspekte der modellgetriebenen Softwareentwicklung auf die Integration im Logistikkontext angewandt werden. Petkoff stellt in [15] einen modellgetriebenen Ansatz zur innerbetrieblichen Integration von Anwendungen vor. Ziel der Arbeit ist das Generieren von Schemata für gängige Datenbanken aus einer Wissensbasis. Der Autor stellt Werkzeuge zur schrittweisen Integration einzelner Anwendungssystem vor und fokussiert die Erstellung von Hilfsmitteln zur Vollintegration von Anwendungssystemen. [15] unterscheidet sich von dem in diesem Beitrag vorgestellten Ansatz neben der reinen Betrachtung der innerbetrieblichen Integration, in den Integrationsarten. Der in diesem Beitrag vorgestellte Ansatz beschreibt die generierte, sofort zur Verfügung stehende Integration mit Hilfe von Human-Service Interaction. Thränert und Kühne wenden in [18] modellgetriebene Konzepte auf das Integration Engineering an und führen den Begriff des Model-Driven Integration Engineering ein. Diese Arbeit legt die theoretischen Grundlagen für den in diesem Beitrag vorgestellten Ansatz und zeigt wie der Einsatz von modellgetriebenen Ansätzen für Integrationsproblemstellungen genutzt werden kann. Als grundlegende Arbeiten dieses Beitrags für den Bereich Human-Service Interaction sind die Arbeiten von Bowen [4], Shneiderman [16] und Dix [6] zu nennen. Von den Autoren wird die menschliche Interaktion mit Computern und Service Systemen betrachtet.

5 Ausblick

In den nächsten Schritten der Forschungsarbeit sollen die Integrationsvarianten II und III näher betrachtet werden. Hierbei sollen semantische Datenintegrationsverfahren zum Mapping der Datenformate der LSEM-Plattform und der Anwendungssysteme betrachtet werden. Außerdem soll die Sicherheit der zu übertragenden Daten im unternehmensübergreifenden Einsatz betrachtet werden und in einer Lösungskomponente von der Integrationsumgebung bereitgestellt werden. Weitere zu bearbeitende Themen stellen somit die Transparenz des Datenverkehrs, die Datenhoheit auf Seiten der LD sowie Sicherheitsmechanismen dar. In einem weiteren Schritt soll das Konzept anhand von Fallstudien validiert und auf Praxistauglichkeit überprüft werden. Bei dieser Überprüfung sollen zeitliche und aufwandsmäßige Kenngrößen zur Bewertung der verschiedenen Integrationsansätze ermittelt werden. Schließlich soll der vorgestellte Ansatz durch eine geeignete Werkzeugunterstützung, zur Erstellung und Transformation von Modellen sowie zur Generierung von Code, erweitert werden.

Die Autoren bedanken sich für die Unterstützung durch das Bundesministerium für Bildung und Forschung (BMBF), welches im Rahmen der Förderprojekte Logistik Service Bus (BMBF 03IP504) und InterLogGrid (BMBF 01IG09010F) die Erarbeitung dieses Beitrags gefördert hat.

6 Literatur

[1] Arsanjani, A, et al. (2007): S3: A Service-Oriented Reference Architecture. IT Professional 9:10-17.

[2] Bauknight, D, Miller, J (1999): Fourth Party Logistics: The Evolution of Supply Chain Outsourcing. In: CALM Supply Chain & Logistics Journal.

[3] Benatallah, B, et al (2005): Developing Adapters for Web Services Integration. 17th International Conference Advanced Information Systems Engineering. Porto.

[4] Bowen, D (1986): Managing customers as human resources in service organizations. In: Human Resource Management 25(3): 371-383, Wilmington.

[5] Bussler, C (2003): B2B Integration: Concepts and Architecture. Springer, Berlin.

[6] Dix, A, et al. (2003): Human-Computer Interaction. Prentice Hall, Harlow.

[7] Erl, T (2009): SOA Design Patterns, Prentice Hall, Upper Saddler River.

[8] Franczyk, B (2010): Logistik-Service-Bus - Projekt. http://www.lsb-plattform.de. Abgerufen am 22.09.2011.

[9] Gallas, B (2007): Enterprise Service Integration (ESI) - Der Weg zur einem servicebasierten EAI-Framework unter Einsatz und Erweiterung von Web Services. In: Aier, S; Schönherr, M. (Hrsg.), Enterprise Application Integration - Flexibilisierung komplexer Unternehmensarchitekturen, Gito-Verl, Berlin.

[10] Hohpe, G, Woolf, B, Brown, K (2010): Enterprise integration patterns: Designing, building, and deploying messaging solutions, Addison-Wesley, Boston.

[11] Klinkmüller, C, et al (2011): The Logistics Service Engineering & Management Platform: Operations, Architecture, Implementation. In: 14th International Conference on Business Information Systems, Poznań, 2011.

[12] Kohlborn, T, Korthaus, A, Rosemann, M (2009): Business and software service lifecycle management. In: IEEE International Conference on Enterprise Distributed Object Computing, Auckland, New Zealand.

[13] Kunkel, R, et al (2011): Modellgetriebene Integration von Logistik-Informationssystemen in die LSEM-Plattform. 41. GI-Jahrestagung Informatik 2011, Die Rolle von Plattformen für Unternehmensökosysteme, Berlin, 2011.

[14] Luo, Z (2010): Service science and logistics informatics. Innovative perspectives. Premier reference source, Business Science Reference, Hershey Pa.

[15] Petkoff, B (2007): Model Driven Application Integration am Beispiel der Versicherungswirtschaft. In: Aier, S; Schönherr, M. (Hrsg.), Enterprise Application Integration - Flexibilisierung komplexer Unternehmensarchitekturen, Gito-Verl, Berlin.

[16] Shneiderman, B (1997): Designing the User Interface: Strategies for Effective Human-Computer Interaction, Addison-Wesley Longman Publishing Co., Boston.

[17] Stahl, T, et.al. (2007): Modellgetriebene Softwareentwicklung. Techniken, Engineering, Management. dpunkt Verlag, Heidelberg.

[18] Thränert, M, Kühne, S (2008): Model-Driven Integration Engineering zur Integration betrieblicher Anwendungssysteme. In: Fähnrich, K; Kühne, S.; Thränert, M. (Hrsg.), Model-Driven Integration Engineering. Universität Leipzig Pressestelle, Leipzig.

[19] Weerawarana, S, Meredith, G, Curbera, F, Christensen, E (2001): Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl. Abgerufen am 22.09.2011.